Project 9: You are to implement the Hough Transform for line detection algorithm.

Abstract: Hough Transform is the best Line Detection Algorithm (self-driving cars use it). There are two spaces used in Hough Transform: a 2D image space and a 2D Hough Space. In the 2D image space, one axis is Row (runs from 0 to numRows -1), the other axis is Col (run from 0 to numCols - 1). Each “cell” of the image space represents a single pixel. In the Hough Space, one axis is Angle (the angle of lines, runs from 0 to 179) and the other axis is Dist (the “orthogonal distance” from the origin to the lines, runs from 0 to diagonal of the image times 2). Each “cell” of Hough Space represents a “line” (with the count of co-linear object pixels on the image space.)

Hough Transform is a mapping from “lines” (on the image) to points (on the Hough Space). All object pixels located on the same line (i.e. co-linear pixels) are mapped (voted) onto a single bucket [angle, distance] in the Hough Space, where angle is the angle of the line passing thru these co-linear pixels and distance is the orthogonal distance from the origin to that line.

\*\*\*\* Remark: Please spend at least the course allotted hours to study the lecture notes on the Blackboard on this topic. The lecture notes are presented in great detail, many thanks to our TA! It includes the theory behind the idea, the mathematic diagrams and equations used, the object-process diagram, the algorithm steps, and many input/output examples. Please study the notes to appreciate this great idea of Dr. Hough who invented the Hough transform for line detection. His idea has been expanded to detection other object shapes.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Language: Java

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Points: 10 pts

Due Date: **Soft copy and pdf hard copies**: 4/12/2020 Sunday before midnight

1 day late: -1 pt 4/13/2020 Monday before midnight

2 days late: -3 pts 4/14/2020 Tuesday before midnight

-10 pts: after 4/15/2020 Tuesday after midnight

\*\*\* Name your pdf file using the same naming convention as given prior

\*\*\* All on-line submission MUST include Soft copy and pdf hard copy

\*\*\* in the same email with correct file names; otherwise, it would not count as submission.

\*\*\*\* You will be given a few test data; run your program on each test data to produce outFile1 and outFile2;

Then, apply your program in project 1 on outFile2 to produce the histogram; select the threshold value at the deepest concavity point of the histogram; then pretty print the threshold outFile2.)

\*\*\*\* Include in your hard copies:

- cover page

- source code

- for each test data print the following:

- outFile1

- outFile2

- histogram of outFile2

- pretty print of threshold outFile2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

I. inFile(argv[0]): a binary image

II. - outFile1 (args[1]): using prettyPrint for visual

- outFile\_2 (args[2]): // the final result of your HoughAry with header information

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

III. Data structure:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- An ImageProcessing class // friend of Hough Transform class

- numRows (int)

- numCols (int)

- minVal (int)

- maxVal (int)

- imgAry (int \*\*) // needs to dynamically allocate at run time

- methods:

- constructor(...)

- loadImage // load imgAry from input file

- A HoughTransform class

- a xyCoord class

- x (int) // need to convert to double when it is used in distance computation.

- y (int) // need to convert to double when it is used in distance computation.

- point (xyCoord)

- angleInDegree (int)

- angleInRadians (double)

- HoughDist (int) // 2 times of the diagonal of the image

- HoughAngle (int) // 180

- HoughMinVal (int)

- HoughMaxVal (int)

- HoughAry (int \*\*) // a 2D int array size of HoughDist by HoughAngle

- methods:

- constructor(...)

- buildHoughSpace (...) // See algorithm steps below

- computeDistance (point, angleInRadians) // on your own

// use the distance formula given in the Lecture Notes

- determineMinMax (HoughAry) // on your own

// read the entire HoughAry to determine HoughMinVal and HoughMaxVal

- prettyPrint(…) // As in your previous projects

- ary2File (HoughAry, outFile2) // output HoughAry to outFile2

- other methods and/or variables if needed.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IV. main (…) // implementation of the algorithm steps in Lecture notes

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 0: inFile 🡨 open input file from args

outFile1, outFile2 🡨 open from args

numRows, numCols, minVal, maxVal 🡨- read from inFile

HoughAngle 🡨 180

HoughDist 🡨2 \* (the diagonal of the input image)

imgAry 🡨 dynamically allocate

HoughAry 🡨 dynamically allocate HoughAry, size of

HoughDist by HoughAngle and initialize to zero

step 1: loadImage(imgAry, inFile)

Step 2: buildHoughSpace (...)

Step 3: prettyPrint(HoughAry, outFile1)

Step 4: determineMinMax (HoughAry)

Step 5: write HoughDist, HoughAngle, HoughMinVal, HoughMaxVal to outFile2

// as the header of Hough image

step 6: ary2File (HoughAry, outFile2) // output HoughAry to outFile2

Step 7: close all files

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

IV. buildHoughSpace (...)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Step 1: scan imgAry left to right and top to bottom

Using r for rows and c for column

Step 2: imgAry (r, c) 🡨 next pixel

Step 3: if imgAry (r, c) > 0

point 🡨 (r, c)

Step 4: angleInDegree 🡨 0

Step 5: angleInRadians 🡨 angleInDegree / 180.00 \* pi

Step 6: dist 🡨 computeDistance (point, angleInRadians)

// Use the distance formula given in the Lecture Notes

Step 7: distInt 🡨 (int) dist

Step 8: HoughAry[distInt][angleInDegree]++

Step 9: angleInDegree ++

step 10: repeat step 5 to Step 9 while angleInDegree <= 179

Step 11: repeat step 2 to step 10 until all pixels are processed

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

V. computeDistance (point, angleInRadians)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Use the distance formula given in the Lecture Notes

/ Make sure the r- & c- coordinate need to convert to double in computation

// add offset to the computation.